NumLin: Linear Types for Numerical Libraries

Dhruv C. Makwana[1 [orcidID]] and Neelakantan R. Krishnaswami[2 [orcidID]]

[1] dcm41@cam.ac.uk dhruvmakwana.com
[2] Department of Computer Science, University of Cambridge
nk480@cl.cam.ac.uk

Abstract. Briefly summarize the contents of the paper in 150–250 words.

Keywords: numerical, linear, algebra, types, permissions, OCaml

1 Introduction

1.1 Contributions

- We have designed the NumLin programming language
- We illustrate that the design is sensible with many matrix-y examples
- We give a soundness proof for NumLin, using a step-indexed logical relation
- Incredibly simple type inference algorihtm for polymorphic fractional permissions
  - Compare to Bierhof et al's *Fraction Polymorphic Permission Inference*, which uses a fancy dataflow analysis
  - We use exactly the same unification algorithm type polymorphism does
- We have an implementation - compatible with and usable from existing code!

2 Language Overview and Examples

2.1 Short language description

This should be a high-level description of the language. We should aim to spend about a page on this section, highlighting the key properties of NumLin:

- Linearity, obviously
- Fractional permissions: sharing and unsharing
- Dynamic errors: matrix dimensions and unsharing (matrix identity checks)
  - In the related work, we should point to L3 statically tracking pointer identities, but we want to keep the number of type indices under control

2.2 Examples

The more examples, the better. In fact, it's actually impossible to have too many examples – it's okay (indeed, desirable) to spend 5-6 pages on examples.

- Simple: factorial, shows recursion, !-annotations etc.
- Less simple: summing over an array, indexing and safety
- Medium: one-dimensioal convolution, permissions
- Harder: linear regression, pattern-matching and apparent non-linearity
- Harder: L1 norm-minimisation, some frees, re-using memory
- Big finish: Kalman filter

3 Formal System

3.1 The Core Type Theory and Dynamic Semantics

Describe the typing rules and operational semantics here

3.2 The logical relation

Describe the step-indexed logical relation and its main properties

3.3 Soundness Theorem

State the fundamental lemma, and sketch the proof a little

4 Implementation

4.1 Implementation Strategy

Talk about how you implemented NumLin and the general architecture. Talk about how simple everything is, and also about how implementing inference for fractions is.

4.2 Performance Metrics

Here, evaluate the performance of the examples from the second section. Compare with your C implementations, and perhaps as well as the straightforward math transcribed into (Matlab/R/Numpy?).

5 Discussion and Related Work

The main point we want to make is that using linear types for BLAS is an "obvious" idea, but is surprisingly under-explored.

- Rust
- ATS
- Single-assignment C
- Linear Haskell
- Bernardy and Sveningsson
- L3
- Boyland fractional permissions

References